МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА»

Лабораторная работа №4

Расширение функциональности пакета для работы с функциями одной переменной

Выполнила:

Иванова Елизавета Вадимовна,
студентка группы 6301-030301D

Самара 2025

Задание 1

Цель: добавить в классы ArrayTabulatedFunction и LinkedListTabulatedFunction конструкторы, принимающие массив объектов типа FunctionPoint, и обеспечить проверку корректности входных данных.

1. В класс ArrayTabulatedFunction добавлен конструктор:

```
public class ArrayTabulatedFunction implements TabulatedFunction,
Serializable {
    private FunctionPoint[] points;
    private int pointsCount;

    public ArrayTabulatedFunction(double leftX, double rightX, int
pointsCount) throws IllegalArgumentException {
        if (leftX >= rightX)
            throw new IllegalArgumentException(" Incorrect Borders");
        if (pointsCount < 2)
            throw new IllegalArgumentException(" Small number of points");
        points = new FunctionPoint[pointsCount + 10];
        this.pointsCount = pointsCount;
        double interval = (rightX - leftX) / (pointsCount - 1);
        for (int i = 0; i < pointsCount; ++i) {
            points[i] = new FunctionPoint(leftX + interval * i, 0.0);
        }
    }
```

2. Аналогично реализован конструктор в LinkedListTabulatedFunction.
3. Для обеспечения инкапсуляции:
    - Созданы копии массива при инициализации.
    - Публичные методы доступа к внутренним данным не предоставляют прямой ссылки на массив/список.
4. Добавлены unit-тесты:
    - Проверка выброса IllegalArgumentException при количестве точек < 2.
    - Проверка выброса исключения при неупорядоченных точках.
    - Проверка корректного создания объекта при правильных данных.

Задание 2

Цель: создать интерфейс Function, содержащий базовые методы для работы с функциями одной переменной, и сделать TabulatedFunction его расширением.

1. В пакете functions создан интерфейс Function:

```
package functions;

public interface Function {
    public double getLeftDomainBorder();
    public double getRightDomainBorder();
    public double getFunctionValue(double x);
}
```

2. Интерфейс TabulatedFunction теперь расширяет Function
3. Удалены дублирующиеся методы getLeftDomainBorder(), getRightDomainBorder(), getFunctionValue() из TabulatedFunction.
4. Все классы, реализующие TabulatedFunction (например, ArrayTabulatedFunction, LinkedListTabulatedFunction), автоматически получили обязанность реализовать эти методы (если они ещё не были реализованы).

Задание 3

Цель: создать пакет functions.basic с классами аналитических функций: Exp, Log, Sin, Cos, Tan.

Созданы классы:

- Exp: возвращает Math.exp(x), границы [-∞, +∞].
- Log: принимает основание, вычисляет log_base(x), проверяет корректность основания.
- TrigonometricFunction: абстрактный класс с общими границами [-∞, +∞].
- Sin, Cos, Tan: наследуют TrigonometricFunction, используют Math.sin, Math.cos, Math.tan.

Задание 4

Цель: создать пакет functions.meta с классами для комбинирования функций: Sum, Mult, Power, Scale, Shift, Composition.

Созданы классы, реализующие Function, с соответствующими вычислениями:

- Sum: возвращает f1(x) + f2(x), область — пересечение.
- Scale: масштабирует x и y вдоль осей.
- Composition: возвращает f1(f2(x)).

Задание 5

Цель: создать класс Functions с фабричными методами для создания комбинированных функций.

Создан класс Functions с приватным конструктором:

```
package functions;

import functions.meta.*;

public class Functions {
    private Functions() {}
    public static Function shift(Function f, double shiftX, double shiftY) {
        return new Shift(f, shiftX, shiftY);
    }
    public static Function scale(Function f, double scaleX, double scaleY) {
        return new Scale(f, scaleX, scaleY);
    }
    public static Function power(Function f, double power) {
        return new Power(f, power);
    }
    public static Function sum(Function f1, Function f2) {
```

```
        return new Sum(f1, f2);
    }
    public static Function mult(Function f1, Function f2) {
        return new Mult(f1, f2);
    }
    public static Function composition(Function f1, Function f2) {
        return new Composition(f1, f2);
    }
}
```

Задание 6

Цель: Создать класс TabulatedFunctions с методом tabulate для создания табулированной функции из аналитической.

Создан метод tabulate:

```
public static TabulatedFunction tabulate(Function function, double leftX,
double rightX, int pointsCount)
        throws IllegalArgumentException {
    if (function.getLeftDomainBorder() > leftX ||
function.getRightDomainBorder() < rightX)
        throw new IllegalArgumentException(" Incorrect Borders");
    FunctionPoint[] points = new FunctionPoint[pointsCount];
    double interval = (rightX - leftX) / (pointsCount - 1);
    for (int i = 0; i < pointsCount; ++i)
        points[i] = new FunctionPoint(leftX + interval * i,
function.getFunctionValue(leftX + interval * i));
    return new LinkedListTabulatedFunction(points);
}
```

Задание 7

Цель: реализовать методы ввода/вывода табулированных функций в байтовые и символьные потоки.

Реализованы методы:

- writeTabulatedFunction: в символьный поток, формат: кол-во точек, x y, x y...
- readTabulatedFunction: из символьного потока с помощью StreamTokenizer.
- outputTabulatedFunction: в байтовый поток с помощью DataOutputStream.
- inputTabulatedFunction: из байтового потока с помощью DataInputStream.

Потоки закрываются с помощью try-with-resources.

Результат:
Методы корректно записывают и считывают табулированные функции. Бинарный формат компактнее, текстовый — читаем.

Задание 8

Создан класс Main, в котором:

- Создаются объекты Sin и Cos, выводятся их значения на отрезке [0, π].

- Создаются табулированные аналоги функций с помощью TabulatedFunctions.tabulate.
- Создаётся функция $\sin^2(x) + \cos^2(x)$ с помощью Functions.power и Functions.sum.
- Табулированная экспонента записывается в exp.txt, затем читается обратно.
- Табулированный логарифм записывается в log_output.bin, затем читается обратно.
- Проверяется, что значения до и после сохранения совпадают.

```
Аналитические функции
Границы области определения для Sin: [-Infinity, Infinity]
Границы области определения для Cos: [-Infinity, Infinity]

Значения аналитических функций на отрезке [0, π]
x        Sin(x)          Cos(x)
-------------------------------------------------
0,0      0,0000000000    1,0000000000
0,1      0,0998334166    0,9950041653
0,2      0,1986693308    0,9800665778
0,3      0,2955202067    0,9553364891
0,4      0,3894183423    0,9210609940
0,5      0,4794255386    0,8775825619
0,6      0,5646424734    0,8253356149
0,7      0,6442176872    0,7648421873
0,8      0,7173560909    0,6967067093
0,9      0,7833269096    0,6216099683
1,0      0,8414709848    0,5403023059
1,1      0,8912073601    0,4535961214
1,2      0,9320390860    0,3623577545
1,3      0,9635581854    0,2674988286
1,4      0,9854497300    0,1699671429
1,5      0,9974949866    0,0707372017
```

```
1,6      0,9995736030    -0,0291995223
1,7      0,9916648105    -0,1288444943
1,8      0,9738476309    -0,2272020947
1,9      0,9463000877    -0,3232895669
2,0      0,9092974268    -0,4161468365
2,1      0,8632093666    -0,5048461046
2,2      0,8084964038    -0,5885011173
2,3      0,7457052122    -0,6662760213
2,4      0,6754631806    -0,7373937155
2,5      0,5984721441    -0,8011436155
2,6      0,5155013718    -0,8568887534
2,7      0,4273798802    -0,9040721420
2,8      0,3349881502    -0,9422223407
2,9      0,2392493292    -0,9709581651
3,0      0,1411200081    -0,9899924966
3,1      0,0415806624    -0,9991351503

Табулированные функции (10 точек на [0, π])
x        TabSin(x)       TabCos(x)
------------------------------------------------
0,0      0,0000000000    1,0000000000
0,1      0,0979815536    0,9827232085
0,2      0,1959631072    0,9654464170
```

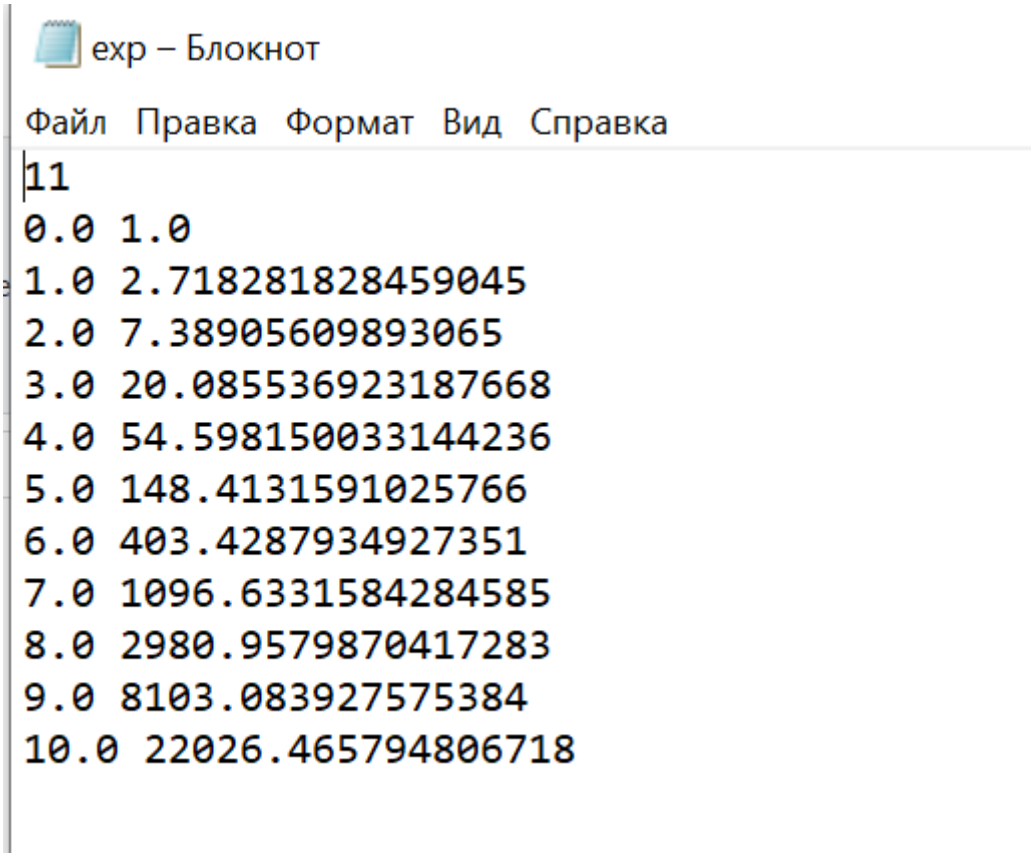
```
0,3     0,2939446608    0,9481696255
0,4     0,3859068057    0,9143546444
0,5     0,4720703379    0,8646081059
0,6     0,5582338701    0,8148615674
0,7     0,6439824415    0,7646204980
0,8     0,7079353587    0,6884043792
0,9     0,7718882758    0,6121882604
1,0     0,8358411930    0,5359721417
1,1     0,8839933571    0,4506334539
1,2     0,9180219936    0,3571405436
1,3     0,9520506300    0,2636476334
1,4     0,9848077530    0,1699305209
1,5     0,9848077530    0,0704374439
1,6     0,9848077530    -0,0290556331
1,7     0,9848077530    -0,1285487101
1,8     0,9662040429    -0,2247614510
1,9     0,9321754065    -0,3182543613
2,0     0,8981467700    -0,4117472716
2,1     0,8624409083    -0,5042718354
2,2     0,7984879911    -0,5804879542
2,3     0,7345350740    -0,6567040730
2,4     0,6705821568    -0,7329201917
2,5     0,5940715695    -0,7941706620
```

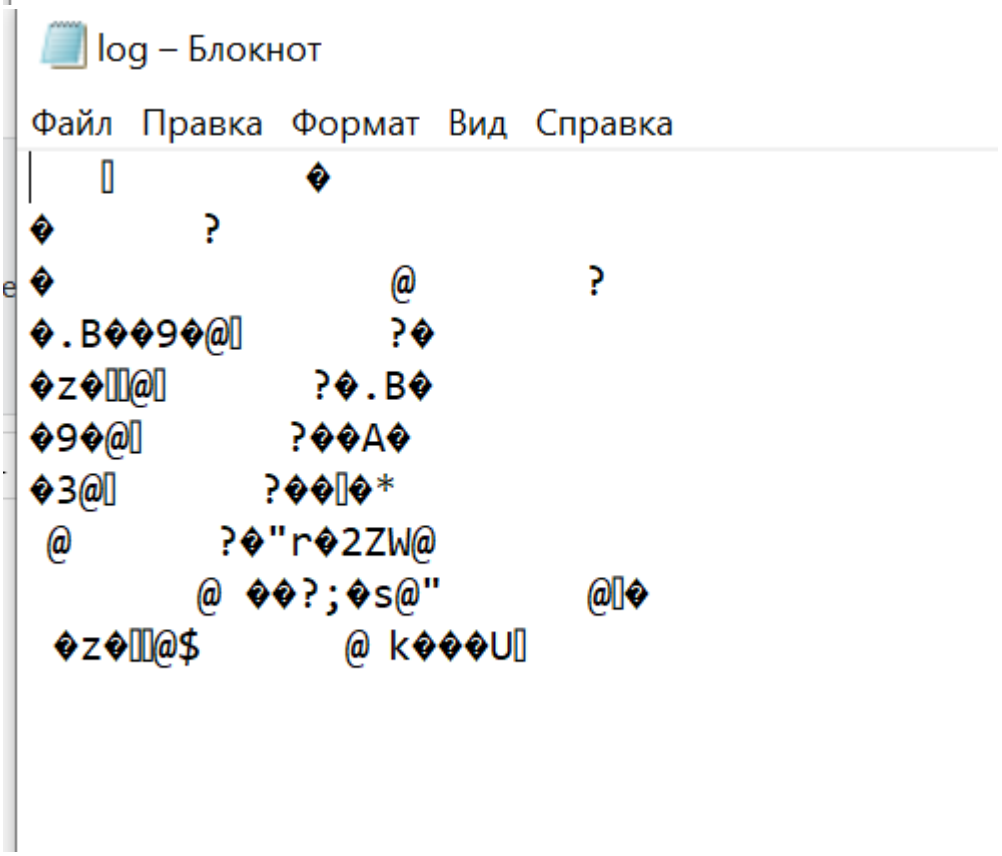
```
Сумма квадратов табулированных функций (sin² + cos²)
x        sin²(x) + cos²(x)
--------------------------------------------------
0,0      1,0000000000
0,1      0,9753452893
0,2      0,9704883234
0,3      0,9854291023
0,4      0,9849684785
0,5      0,9703975808
0,6      0,9756244278
0,7      0,9993578909
0,8      0,9750730614
0,9      0,9705859766
1,0      0,9858966365
1,1      0,9845147652
1,2      0,9703137486
1,3      0,9759104767
1,4      0,9987226923
1,5      0,9748077439
1,6      0,9706905402
1,7      0,9863710813
1,8      0,9840679624
1,9      0,9702368269
```

```
2,6      0,9701668157
2,7      0,9765033061
2,8      0,9974730266
2,9      0,9742978404
3,0      0,9709203989
3,1      0,9873407022

Экспонента (табулированная функция)
Оригинальная табулированная экспонента:
X               Y
--------------------------------------------------
0,000000        1,000000
1,000000        2,718282
2,000000        7,389056
3,000000        20,085537
4,000000        54,598150
5,000000        148,413159
6,000000        403,428793
7,000000        1096,633158
8,000000        2980,957987
9,000000        8103,083928
10,000000       22026,465795
```

```
Экспонента после записи и чтения из текстового файла:
X               Y
--------------------------------------------------
0,000000        1,000000
1,000000        2,718282
2,000000        7,389056
3,000000        20,085537
4,000000        54,598150
5,000000        148,413159
6,000000        403,428793
7,000000        1096,633158
8,000000        2980,957987
9,000000        8103,083928
10,000000       22026,465795

Логарифм (табулированная функция)
Оригинальная табулированная логарифмическая функция:
X               Y
--------------------------------------------------
0,100000        -2,302585
1,090000        0,086178
2,080000        0,732368
```

```
4,060000        1,401183
5,050000        1,619388
6,040000        1,798404
7,030000        1,950187
8,020000        2,081938
9,010000        2,198335
10,000000       2,302585

Логарифм после записи и чтения из бинарного файла:
X               Y
--------------------------------------------------
0,100000        -2,302585
1,090000        0,086178
2,080000        0,732368
3,070000        1,121678
4,060000        1,401183
5,050000        1,619388
6,040000        1,798404
7,030000        1,950187
8,020000        2,081938
9,010000        2,198335
10,000000       2,302585
```





Файл log_output.bin был создан с помощью метода TabulatedFunctions.outputTabulatedFunction(...), который использует DataOutputStream. DataOutputStream записывает данные в бинарном формате — то есть числа записываются в виде байтов, а не символов. Когда я открываю такой файл в Блокноте, он пытается интерпретировать эти байты как текст. Но бинарные данные не являются текстом — поэтому Блокнот показывает мусор: NUL, SOH, DC1, ?, @ и т.д.

Задание 9

Цель: сделать объекты TabulatedFunction сериализуемыми с помощью Serializable и Externalizable.

1. ArrayTabulatedFunction реализует Serializable
2. ArrayTabulatedFunction реализует Externalizable
3. Тестирование: табулированная функция сериализуется в serialized_func.ser, затем десериализуется.

```
Сериализация ArrayTabulatedFunction
Сериализация и десериализация ArrayTabulatedFunction успешна:
X               Y
-------------------------------------------------
0,000000        0,000000
2,500000        0,000000
5,000000        0,000000
7,500000        0,000000
10,000000       0,000000

Process finished with exit code 0
```

Результат:
Оба способа сериализации работают. Externalizable позволяет контролировать формат, Serializable — проще в реализации.